정 보 보 호 학 회 지
제 30 권 제3호, 2020. 6

랜섬웨어 암호기능 및 복구 가능성 분석

이 영 주*

요 약

2019년에는 기존 랜섬웨어의 변형된 형태 또는 새롭게 개발된 형태의 랜섬웨어 공격이 전 세계적으로 발생했다. 공격에 따른 금전적 피해의 증가를 방지하기 위해 랜섬웨어의 파일 암호화 과정의 암호기능을 분석하여 복구 가능성을 판단할 필요가 있다. 본 논문에서는 2019년 한 해 발생한 다양한 랜섬웨어의 실행과정, 암호화 과정, 키 생성 과정 등을 분석하여 복구 가능성을 판단하고자 한다.

Ⅰ. 서 론

작년과 마찬가지로 2020년에도 랜섬웨어 공격이 지속해서 발생하고 있다. 다만, 과거에는 주로 불특정 개인 PC를 대상으로 무차별 감염을 시도하는 공격이 발생했는데, 점차 공공기관과 기업을 대상으로 APT를 통해 랜섬웨어를 감염시키는 공격이 증가하고 있다[1]. 랜섬웨어는 예방하는 것이 최고의 방법이지만, 최근 등장한 랜섬웨어는 새로운 공격 방법을 사용하기 때문에 개선된 네트워크 및 시스템 모니터링 보안제품을 사용하지 않는다면 탐지하기 어렵다. 따라서 랜섬웨어 감염에 대한 사후대응의 목적으로 랜섬웨어의 파일 암호화 과정의 암호기능을 분석하여 복구 가능성을 판단할 필요가 있다. 복구 가능성이 있는 경우 암호 키 재현을 통해 랜섬웨어 복구도구를 개발하여 감염자에게 제공함으로써 피해가 확산하는 것을 방지해야 한다.

이에 따라, 국내외 공공기관과 보안업체는 랜섬웨어의 파일 암호화 시 암호기능을 분석하여 해당 랜섬웨어가 복구 가능한지 확인하고 있다, 그리고, 분석결과를 토대로 개발된 랜섬웨어 복구도구는 유로폴(Europol)에서 유지 관리하는 노모어랜섬(NoMoreRansom) 사이트에서 무료로 배포 중이다[2]. 따라서 본 논문에서는 국내외 랜섬웨어 대응 동향에 따라 랜섬웨어의 암호기능 분석을 통해 복구 가능성을 판단하고자 한다.

Ⅱ. 랜섬웨어 암호기능 및 복구가능성 분석

본 장에서는 랜섬웨어의 실행과정과 암호화 과정을 분석하고, 결과를 기반으로 복구 가능성을 판단한다. 복구 가능한 랜섬웨어의 경우, 복구 방법에 대해서 알아보고자 한다.

2.1. CLOP

CLOP 랜섬웨어는 2019년 2월경부터 국내 기업용 중앙관리 서버인 AD(Active Directory) 서버를 대상으로 피해를 주기 시작한 랜섬웨어이다[3]. 해당 랜섬웨어는 AD 서버를 감염시킨 후에, 네트워크 드라이브를 탐색하여 서버에 연결된 모든 드라이브를 암호화한다[4]. 분석에 사용한 CLOP 랜섬웨어 해시 값은 [표 1]과 같다.

〔표 1〕 CLOP 랜섬웨어 분석 샘플 해시 값

MD5	846f93fcb65c9e01d99b867fea384edc
SHA1	d17a7b14689b5cd3ed23432e3d99abdc2e2544ae
SHA256	87a0d69701a2ca627a0280115681f49f66b227f3d0f32b155f73969062567137

2.1.1. 실행과정

CLOP 랜섬웨어는 감염 대상 PC에서 실행된 후, 시스템이 사용하는 언어를 검사한다. 러시아어 환경이면 감염시키지 않고 CLOP 랜섬웨어 실행 파일을 삭제한

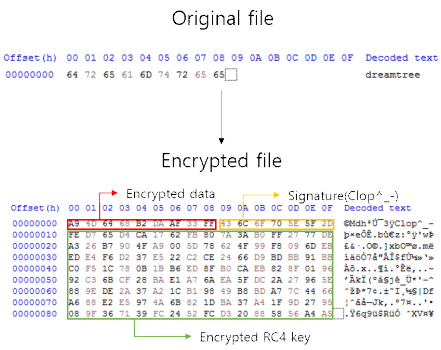
* 한국인터넷진흥원 (주임연구원, lyj5607@kisa.or.kr)

후 실행 과정을 종료한다. 러시아어 환경이 아닌 경우 리소스 영역에 암호화된 상태로 존재하는 배치파을 복호화한다. 복호화된 배치파일은 시스템 복구를 할 수 없도록 볼륨 섀도 복사본 파일을 삭제한다. 그리고 문서나 데이터베이스 등의 프로그램을 실행하는 프로세스들을 강제로 종료한다. 해당 랜섬웨어는 파일 암호화 시 디렉터리마다 랜섬노트를 생성한다[5].

2.1.2. 암호화 과정

CLOP 랜섬웨어는 파일 암호화 시 사용하는 RC4 암호 키를 CryptGenKey 함수를 사용하여 생성한다. 모든 파일에 대하여 다른 암호 키를 사용하며, 암호 키 생성 과정은 모두 같다. 원본 파일의 크기가 300,000 bytes 미만인 경우, 파일 전체를 대상으로 암호화하고, 반대의 경우에는 원본 파일의 상위 300,000 bytes 크기만 암호화한다.

CLOP 랜섬웨어는 파일 암호화가 완료되면 사용한 RC4 암호 키를 RSA-1024 공개 키로 암호화한다. 공개 키는 CLOP 랜섬웨어 내에 하드코딩되어 있다. 암호화된 파일의 구조는 [그림 1]과 같다. RC4 암호 키로 암호화한 데이터 뒤에 'Clop^_-' 문자열이 위치한다. 마지막 128 bytes 크기의 데이터는 RC4 암호 키를 RSA-1024 공개 키로 암호화한 데이터이다[6].



〔그림 1〕 암호화된 파일 구조

2.1.3 복구 가능성

CLOP 랜섬웨어는 파일 암호화 시 RC4 스트림 암호를 사용하며, CryptoAPI 에서 제공하는 CryptGenKey 함수를 사용하여 랜덤한 암호 키를 생성한다. 그리고 사용된 RC4 암호 키는 공격자의 RSA-1024 공개 키로 암호화한다. 그러므로 공격자의 공개 키에 대응하는 개인 키를 획득할 수 있는 경우에만 암호화된 RC4 암호 키를 복호화하여 파일 복호화 시 사용할 수 있다.

그리고 파일 암호화에 사용된 암호 키들은 사용이 끝나면 메모리 내에서 삭제되므로 키 정보를 추출하는 것은 불가능하다. 볼륨 섀도 복사본을 삭제하기 때문에 시스템 백업 복구 방법을 통한 파일 복구도 불가능하다〔7〕.

2.2. LooCipher

2019년 6월 말에 발견된 LooCipher 랜섬웨어는 스팸메일에 첨부된 악성 워드 파일인 'Info_BSV_2019.docm'을 통해서 감염된다. PC 사용자가 해당 파일을 실행하면 워드 문서의 매크로 기능을 활성화하도록 요구한다. 매크로를 활성화하면 매크로 스크립트가 동작하여 자동으로 'LooCipher.exe' 파일을 다운로드하고 실행한다〔8〕. 분석에 사용한 LooCipher 랜섬웨어 해시 값은 〔표 2〕와 같다.

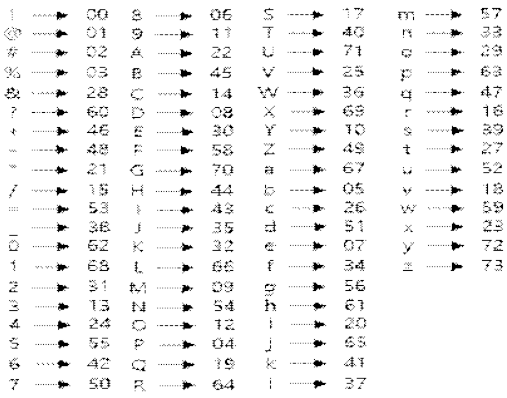
〔표 2〕 LooCipher 랜섬웨어 분석 샘플 해시 값

MD5	a0609d7ad40461dab889944bfe8ca588
SHA1	939e84218cd1116b10166ed8352c11ad16cc2585
SHA256	924cc338d5d03f8914fe54f184596415563c4172679a950245ac94c80c023c7d

2.2.1. 실행과정

LooCipher 랜섬웨어는 감염 대상 PC에서 실행된 후, 파일 암호화를 수행한다. 암호화된 파일의 확장자 뒤에 '.lcphr' 확장자가 추가된다. 해당 랜섬웨어는 암호화 시 새로운 파일을 생성하여 암호화한 파일의 데이터를 저장하고, 원본 파일의 데이터는 삭제한다. 이때, 원본 파일은 삭제하지 않고 0KB 상태로 남겨둔다.

사용한 16 bytes 크기의 암호 키는 [그림 2]와 같이 1 byte 문자마다 고정된 2자리 숫자로 대응되어 인코딩된 후 C&C 서버로 전송된다. 암호 키 전송 후에는 감



〔그림 2〕 암호 키 인코딩 규칙

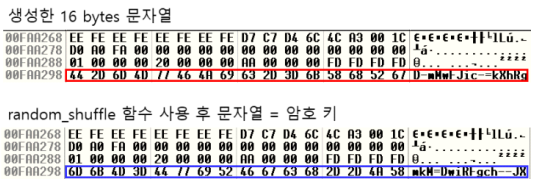
염된 PC 바탕화면에 랜섬노트가 생성된다[9].

2.2.2. 암호화 과정

LooCipher 랜섬웨어는 파일 암호화에 사용하는 16 bytes 크기의 암호 키를 생성하기 위해 시스템 시간을 seed로 사용하는 rand 함수를 사용하여 난수를 생성한다. 그리고 생성한 난수를 0x49 값으로 나눈 나머지 값을 계산한 후, 특정 문자열(!@#%&?+-*/=_0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz)에서 앞서 계산한 나머지 값에 해당하는 위치의 문자를 선택한다. 해당 과정은 16회 반복되어 16 bytes 크기의 문자열이 생성된다.

마지막으로, C++ 라이브러리의 random_shuffle 함수를 호출한다. random_shuffle 함수는 random generator에서 생성한 seed를 사용하여 배열의 값들을 랜덤하게 재배치하는 기능을 수행한다. LooCipher 랜섬웨어는 [그림 3]과 같이 random_shuffle 함수를 사용하여 랜덤하게 재배치한 값을 AES-128 암호 키로 사용한다.

LooCipher 랜섬웨어는 파일 암호화 시 Crypto++의 AES-128-ECB 암호 알고리즘을 사용하며, 모든 파일에



〔그림 3〕 최종적으로 생성한 AES-128 암호 키

대해 16 bytes 크기의 암호 키를 같게 사용한다. 암호화가 완료되면 사용된 암호 키는 인코딩되어 C&C 서버로 전송되며, 해당 키는 랜섬웨어 실행이 완료된 이후에도 메모리에 존재한다[10].

2.2.3. 복구 가능성

LooCipher 랜섬웨어는 감염 당시의 시스템 시간을 seed로 사용하는 rand 함수를 이용해 암호 키를 생성하여 파일을 암호화한다. 따라서 암호화된 파일의 수정 시간을 기준으로 전후 시간을 rand 함수에 입력하여 암호 키 후보군을 생성하는 방법을 통해 암호 키를 복구할 수 있다. 암호 키 후보군을 사용해 암호화된 파일에 대해 복호화를 수행하고, 복호화한 결과가 해당 원본 파일의 고유한 시그니처 값을 가진 경우 복호화가 성공한 것이다. 해당 암호 키를 사용하면 모든 파일에 대해 복호화를 수행할 수 있다.

또한, LooCipher 랜섬웨어는 C&C 서버에 암호 키를 전송하는 과정에서 메모리 내 암호 키를 삭제하거나 제로화하지 않는다. 따라서 해당 메모리 영역이 덮어쓰기 되지 않았다면 메모리 정보 추출을 통해 암호 키를 복구할 수 있다[11].

2.3. Phobos

2019년 1월경에 발견된 Phobos 랜섬웨어는 RDP(Remote Desktop Protocol)의 취약한 비밀번호 사용자들을 노려 사전공격 및 무작위 대입 방식을 통해 PC에 접근한 후, 사용자 몰래 악성 파일을 실행시키는 방법으로 유포됐다[12]. 2016년 발견된 Dharma와 Crysis 랜섬웨어의 변종이다[13]. 분석에 사용한 Phobos 랜섬웨어 해시 값은 [표 3]과 같다.

〔표 3〕 Phobos 랜섬웨어 분석 샘플 해시 값

MD5	ce847b271ecc1e3e47a30b10f96bb182
SHA1	f607a5b0cfe604402edde6c02ae0dd447c594381
SHA256	8e988daebd0bcbfa79dacab92930e6b061cb8c0f7b8afbecc1583ab5e987c0cd

2.3.1. 실행과정

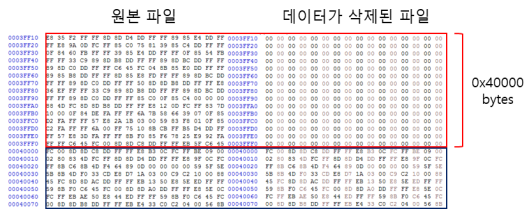
Phobos 랜섬웨어는 PC에서 실행된 후 시스템이 사용하는 언어를 검사하여 감염 대상인지 확인한다. 러시아어 사용 환경이면 랜섬웨어는 실행되지 않고 종료된다. 러시아어 사용 환경이 아닌 경우에는 볼륨 시리얼 정보를 이용하여 뮤텍스를 생성하고, 랜섬웨어 실행 프로그램의 권한을 관리자 권한으로 상승시켜 해당 프로그램을 새로운 프로세스에 할당하여 실행한다.

‘%APPDATA%\Local’ 경로에는 Phobos 랜섬웨어 실행 프로그램이 복제된다. 복제된 실행 프로그램은 레지스트리 경로(HKLM\MicroSoft\Windows\CurrentVersion\Run)에 등록되어 시스템이 실행될 때마다 해당 랜섬웨어는 자동으로 실행된다. 그리고 감염자가 윈도우의 시스템 복원 기능을 사용하지 못하도록 볼륨 섀도 복사본을 삭제하고, 네트워크 방화벽을 해제한다. 파일 암호화 시 방해되는 데이터베이스나 문서와 관련된 프로세스들을 강제 종료한다. 마지막으로 파일 암호화를 수행한 후 랜섬노트를 생성한다[14].

2.3.2. 암호화 과정

Phobos 랜섬웨어는 로컬 드라이브에 쓰레드 4개, 네트워크 드라이브에 쓰레드 2개를 생성한다. CryptGenRandom 함수를 사용하여 쓰레드마다 32 bytes 크기의 AES 암호 키를 한 번씩만 생성한다. AES 암호 키는 파일 암호화가 완료된 후, 공격자의 RSA-1024 공개 키로 암호화된다. 이때, Windows API에서 제공하는 RSA 암호 알고리즘 대신 Github에서 제공하는 RSA 암호 알고리즘이 사용된다. AES 암호 키 생성 후에 CryptGenRandom 함수를 사용하여 16 bytes 크기의 IV를 생성하고, 사용 후에 암호화된 파일에 저장한다.

Phobos 랜섬웨어는 앞서 생성한 IV 값과 AES 암호



〔그림 4〕 원본 파일(좌), 데이터가 삭제된 파일(우)

키를 사용하여 AES-256-CBC 암호 알고리즘으로 파일을 암호화하며, CryptoAPI의 CryptEncrypt 함수를 사용한다. 파일의 크기가 0x180000 bytes 미만이면 암호화하며, 로컬 드라이브뿐만 아니라 네트워크 드라이브도 탐색하여 암호화를 수행한다. 파일의 크기가 0x180000 bytes 이상일 경우 [그림 4]와 같이 상위 0x40000 bytes에 해당하는 데이터를 삭제하여 데이터 복원을 어렵게 하고, 파일은 암호화하지 않는다.

AES 암호 키 사용이 완료되면, 키 삭제 기능을 수행하는 CryptDestroyKey 함수를 사용하여 메모리에서 AES 암호 키를 삭제한다. 그리고 Heap 메모리 영역에 할당된 데이터를 해제하는 기능을 수행하는 HeapFree 함수를 사용하여 메모리에서 AES 암호 키 정보를 삭제한다[15].

2.3.3. 복구 가능성

Phobos 랜섬웨어는 파일 암호화 시 AES-256-CBC 암호 알고리즘을 사용하며, CryptoAPI의 CryptGenRandom 함수를 사용하여 랜덤하게 암호 키를 생성한다. 사용이 완료된 AES 암호 키는 공격자의 RSA-1024 공개 키로 암호화한다. 그러므로 공격자의 공개 키에 대응하는 개인 키를 획득할 수 있는 경우에만 암호화된 AES 암호 키를 복호화하여 파일 복호화 시 사용할 수 있다. 하지만 파일의 크기가 0x180000 bytes 이상이면 암호화하지 않고 파일 일부분을 삭제하기 때문에, 공격자의 RSA-1024 개인 키를 획득한 경우라도 복구는 불가능하다.

파일 암호화에 사용한 AES 암호 키는 사용이 완료되면 메모리 내에서 삭제되므로 키 정보를 추출하는 것은 불가능하다. 또한, 볼륨 섀도 복사본을 삭제하기 때문에 시스템 백업 복구 방법을 통한 파일 복구도 불가능하다[16].

2.4. JCry

2019년 3월에 발견된 JCry 랜섬웨어는 Go 언어로 제작되었으며, 이스라엘 웹 사이트의 가짜 어도비 플래시 플레이어 업데이트 메시지를 통해서 유포됐다. 웹 사이트를 방문한 사용자의 PC가 윈도우 환경이라면 가짜 어도비 플래시 플레이어 업데이트 메시지가 나타나며,

이를 클릭하면 ‘flash player_install.exe’ 파일이 실행되면서 JCry 랜섬웨어에 감염된다[17]. 분석에 사용한 JCry 랜섬웨어 해시 값은 [표 4]와 같다.

〔표 4〕 JCry 랜섬웨어 분석 샘플 해시 값

MD5	C86C75804435EFC380D7FC436E344898
SHA1	9AAB879DB9AA96683FEB1BE7F741AFAF7099C665
SHA256	D7E118A3753A132FBEDD262FDF4809A76CE121F758EB6C829D9C5DE1FFAB5A3B

2.4.1. 실행과정

JCry 랜섬웨어가 실행되면 ‘%ROAMING%\Microsoft\Windows\StartMenu\Programs\Startup’ 경로에 ‘Enc.exe’ 파일과 ‘Dec.exe’ 파일이 저장된다. ‘Enc.exe’ 파일은 파일 암호화 시 사용되며, ‘Dec.exe’ 파일은 감염자가 공격자에게 복구 비용을 지급할 경우 암호화된 파일을 복호화하는 데 사용된다.

파일 생성 후 파일 암호화 시 사용하는 16 bytes 크기의 암호 키 1개를 생성하고, 감염 PC의 Startup 폴더에 ‘PersonalKey.txt’ 파일이 존재하는지 확인한다. 만약 해당 파일이 존재하면 이미 감염된 PC라고 판단하여 랜섬웨어 실행을 종료하며, 반대의 경우 파일 암호화를 진행한다. 암호화된 파일에는 기존의 확장자에 ‘.jcry’ 확장자가 추가되며, 원본 파일은 삭제된다.

파일 암호화가 완료되면 바탕화면에 HTML 형식의 랜섬노트를 생성한다. 랜섬노트 내에는 파일 암호화에 사용한 암호 키를 암호화한 키 값인 Unique Key가 포함되어 있다. 감염자가 윈도우 백업 기능을 사용하여 데이터를 복구하는 것을 방지하기 위해 커맨드 명령어를 실행하여 볼륨 섀도 복사본을 삭제한다[18].

2.4.2. 암호화 과정

JCry 랜섬웨어는 공격자가 임의로 생성한 main_RandASCIIBytes 함수를 사용하여 16 bytes 크기의 암호 키를 생성한다. 자세한 암호 키 생성과정을 살펴보면 먼저, 함수 내부에서 Windows CryptoAPI의 CryptoAcquireContext 함수를 호출하여 CSP(Cryptographic Service Provider)의 핸들을 생성한다. 그리고 암호학적 난수 발생기인 CryptGenRandom 함수를 사용하여 16 bytes 크기의 난수를 생성하고, 이를 파일 암호 키로 사용한다.

파일 암호 키는 랜섬웨어 코드 내에 하드코딩된 PEM(Privacy Enhanced Mail) 구조의 공격자 공개 키를 사용하여 RSA-4096-OAEP 암호 알고리즘으로 암호화된다. 이때 Go 언어에서 제공하는 crypto/rsa 패키지의 EncryptOAEP 함수를 사용한다. 암호화된 파일 암호 키는 Hex String으로 인코딩된 후 ‘PersonalKey.txt’ 파일에 저장된다.

JCry 랜섬웨어는 파일 암호화 시 AES-128-GCM 암호 알고리즘을 사용하며, 파일 암호화에 사용되는 암호 키와 nonce는 CryptoAPI의 CryptoGenRandom 함수로 생성한 난수를 사용한다. 모든 파일마다 같은 암호 키를 사용하여 암호화를 수행하며, nonce는 파일마다 새롭게 생성한다. 암호화 시 원본 파일의 0x100000 bytes 크기만 암호화하며, 나머지 데이터는 암호화하지 않는다. 암호화된 파일은 상위부터 12 bytes 크기의 nonce, 암호화된 데이터, 16 bytes 크기의 태그 값으로 구성된다.

AES 암호 키 사용이 완료되면, JCry 랜섬웨어는 Go 언어의 메모리 관리 기법인 Garbage Collection을 사용한다. Garbage Collection은 동적 할당된 메모리 영역 중 사용하지 않는 영역을 찾아 자동으로 할당 해제하는 기능을 수행한다. 이 기법에 따라 파일 암호화 시 사용된 암호 키 정보가 삭제된다[19].

2.4.3. 복구 가능성

JCry 랜섬웨어는 파일 암호화 시 AES-128-GCM 암호 알고리즘을 사용하며, CSPRNG(Cryptographically Secure Pseudo Random Number Generator)의 표준 난수 발생기인 CryptoGenRandom 함수를 사용하여 파일 암호 키와 nonce를 생성한다.

암호 키는 공격자의 공개 키를 사용하여 RSA-4096-OAEP로 암호화하기 때문에, 이에 대응하는 개인 키를 획득할 수 있는 경우에만 암호화된 AES 암호 키를 복호화하여 파일 복호화 시 사용할 수 있다. 그리고 Go 언어의 Garbage Collection 기법을 사용하여 암호 키가 저장된 메모리 영역에 대해 할당 해제를 수행하기 때문에, 메모리에서 키 정보를 추출하는 것은 불

가능하다[20].

2.5. Maze

Maze 랜섬웨어는 2019년 5월 14일경 해외에서 발견된 ChaCha 랜섬웨어의 변종으로써 2019년 5월 24일 국내에서 발견됐다[21]. 해당 랜섬웨어는 Windows의 Powershell을 통해 다운로드되며, 악성코드 제작 툴 킷인 Fallout Exploit Kit에 의해 유포됐다[22]. 분석에 사용한 Maze 랜섬웨어 해시 값은 [표 5]와 같다.

〔표 5〕 Maze 랜섬웨어 분석 샘플 해시 값

MD5	9823800F063A1D4EE7A749961DB7540F
SHA1	9D2917A668B30BA9F6B3E7A3316553791EB1C052
SHA256	A9524DE985A3ECC43E11DD7C051A4BBFE08C3D71CDE98EA9BB6EA7F32C0CB174

2.5.1. 실행과정

Maze 랜섬웨어는 감염 PC의 기기 정보를 수집 후에 15개의 C&C 서버 IP 주소로 전송한다. 2019년 10월 기준, 2개의 IP는 차단되었으며, 남은 13개의 IP는 연결은 가능하지만 원활한 통신은 불가능하다. C&C 서버와 통신이 완료되면 윈도우 시스템 복원 기능을 사용한 데이터 복구를 방지하기 위해 볼륨 새도 복사본을 삭제한다. 후에 파일 암호화를 수행하며, 암호화된 파일의 확장자 끝에 4~7자리의 랜덤한 문자열이 추가된다. 그리고 암호화된 파일의 폴더에 html 형식의 랜섬노트를 생성한다[23].

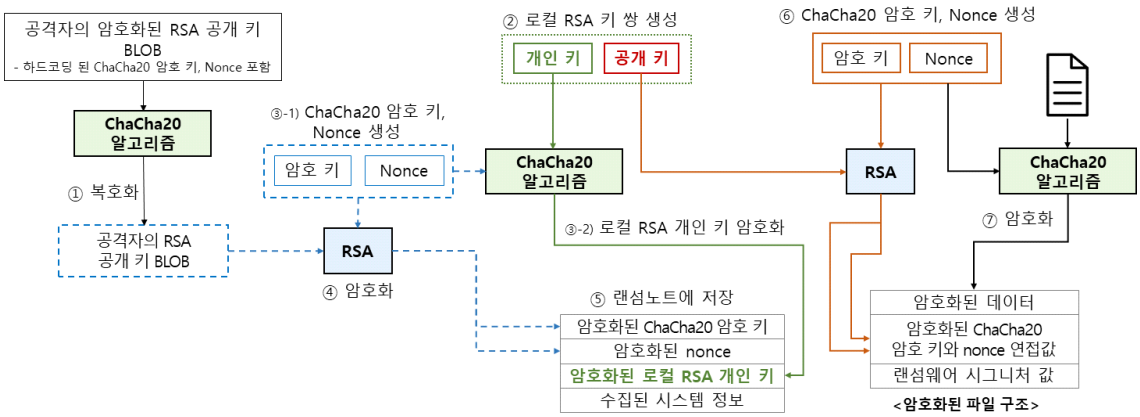
2.5.2. 암호화 과정

Maze 랜섬웨어의 암호 키 생성 과정은 [그림 5]와 같다. 먼저 실행 프로그램의 data 영역에 ChaCha 암호 알고리즘으로 암호화된 공격자의 RSA 공개 키 BLOB을 복호화한다. 복호화 시 필요한 암호 키와 nonce는 data 영역 내에 하드코딩되어 있다.

공격자의 공개 키 BLOB을 복호화한 후, Crypto API에서 제공하는 CryptGenKey 함수를 사용하여 로컬에서 RSA 공개 키와 개인 키를 생성한다. CryptGenRandom 함수를 사용하여 ChaCha20 암호 키와 nonce를 생성하고, 이를 사용하여 로컬에서 생성한 RSA 개인 키를 ChaCha20 암호 알고리즘으로 암호화한다. 이때, 사용된 암호 키와 nonce의 크기는 각각 0x20 bytes, 0x80 bytes이다.

처음에 복호화한 공격자의 RSA 공개 키 BLOB을 CryptImportKey 함수를 사용하여 import한다. 그리고 CryptEncrypt 함수를 사용하여 로컬에서 생성한 RSA 개인 키를 암호화하는 데 사용한 ChaCha20 암호 키와 nonce를 각각 암호화한다. 사용자 ID, OS 정보 등 수집한 시스템 정보를 앞서 암호화한 로컬 RSA 개인 키, ChaCha20 암호 키, nonce와 연접하고 Base64로 인코딩한 후, 결과 값을 랜섬노트에 저장한다.

앞의 과정이 완료되면, Maze 랜섬웨어는



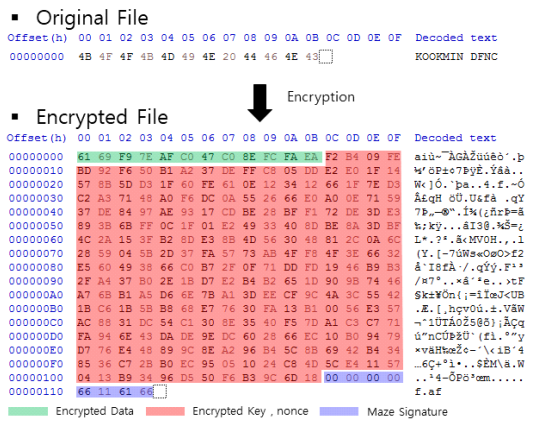
〔그림 5〕 Maze 랜섬웨어 암호 키 생성 과정

CryptGenRandom 함수를 사용하여 파일 암호화 시 사용하는 ChaCha20 암호 키와 nonce를 랜덤하게 생성한다. 이때, 생성된 암호 키와 nonce의 크기는 각각 0x20 bytes, 0x80 bytes이다. 마지막으로, 로컬에서 생성한 RSA 공개 키를 import하여 파일 암호화 시 사용하는 ChaCha20 암호 키와 nonce를 연접한 후 RSA 알고리즘으로 암호화한다.

Maze 랜섬웨어는 로컬에서 랜덤하게 생성한 ChaCha20암호 키와 nonce를 사용하여 파일 암호화를 수행한다. 암호화된 파일의 구조는 [그림 6]과 같다.

암호화된 데이터의 마지막 부분에서부터 0x100 bytes 크기는 파일 암호화 시 사용한 ChaCha20암호 키와 nonce를 연접하여 로컬 RSA 공개 키로 암호화한 값이다. 그다음 8 bytes는 Maze 랜섬웨어의 Signature이다.

Maze 랜섬웨어는 파일 암호화가 완료되면 CryptoAPI에서 제공하는 CryptReleaseContext 함수를 사용하여 메모리에서 키 관련 정보를 삭제한다. 그리고 WindowsAPI에서 제공하는 Virtual 함수를 사용하여 암호 키와 nonce의 해당 메모리 값을 해제한다[24].



〔그림 6〕 암호화된 파일 구조

2.5.3. 복구 가능성

Maze 랜섬웨어는 파일 암호화 시 ChaCha20과 RSA-2048 암호 알고리즘을 사용한다. 파일마다 CryptoAPI의 CryptGenRandom 함수를 사용하여 서로 다른 키와 nonce를 생성한 뒤 파일을 암호화한다. 이때, 사용된 키는 메모리 할당 해제 및 제로화하기 때문에 메모리 분석을 통해 암호 키를 복구할 수 없다.

해당 랜섬웨어에 감염된 파일을 복호화하기 위해서는 암호화된 파일에 저장된 암호화된 ChaCha20 암호 키와 nonce가 필요하다. 이 값들은 로컬에서 생성한 RSA 공개 키로 암호화됐기 때문에, 공개 키에 대응하는 RSA 개인 키를 사용해야 복호화할 수 있다. 하지만 RSA 개인 키는 ChaCha20 암호 알고리즘으로 암호화됐고, 이때 사용한 ChaCha20 암호 키와 nonce는 공격자의 공개 키로 암호화됐다. 따라서 공격자의 공개 키에 대응하는 개인 키를 획득할 수 없다면 암호화된 파일을 복호화하는 것은 불가능하다[25].

2.6. Mgniber2

2017년 10월에 처음 등장한 Magniber 랜섬웨어는 익스플로러 취약점을 악용하여 윈도우에 침투하며, 운영체제 시스템이 한국어 환경인 경우에만 감염된다. 버전 1의 경우, 모든 파일에 대해 같은 암호 키를 사용하며, C&C(Command and Control) 서버를 통해 암호 키를 가져온다. 만약, C&C 서버를 통해 암호 키를 가져오지 못하면, 코드 내에 하드코딩된 암호 키를 사용하여 파일을 암호화한다[26]. 2018년 3월, 안랩에서는 하드코딩된 암호 키를 사용하는 Magniber 랜섬웨어의 복호화 도구를 배포하였다[27].

하지만 2018년 7월, 악성 웹 사이트에 접근하면 광고에 포함된 악성 스크립트가 실행되어 감염되는 멀버타이징 기법을 사용한 Magniber2 랜섬웨어가 등장했다. 이전 버전과 달리, 한국뿐만 아니라 중국, 대만, 홍

〔표 6〕 Magniber2 랜섬웨어 분석 샘플 해시 값

Loader.dll	MD5	72FCE87A976667A8C09ED844564ADC75
	SHA1	D3E17F5ECA5FB23B272549692D84CC449CF71527
	SHA256	6E57159209611F2531104449F4BB86A7621FB9FBC2E90ADD2ECDFBE293AA9DFC
Payload.dll	MD5	19599CAD1BBCA18AC6473E64710443B7
	SHA1	F9E2111E2903838BB9F4EFB557F75745D028BC3E
	SHA256	FB6C80AE783C1881487F2376F5CACE7532C5EADFC170B39E06E17492652581C2

콩 등 아시아 국가들에서도 발견됐다. 또한, 파일리스 기법을 사용하여 감염자 및 백신 프로그램이 탐지하기 어렵다. 분석에 사용한 Magniber2 랜섬웨어의 샘플 해시값은 [표 6]과 같다.

Magniber2 랜섬웨어는 파일리스 형태로 실행되어 별도의 실행 파일이 없으므로, 프로세스 메모리에서 DLL 파일을 읽어와 랜섬웨어 동작을 수행한다. Loader.dll 파일과 Payload.dll 파일에 대한 설명은 실행과정에서 자세하게 살펴본다.

2.6.1. 실행과정

Magniber2 랜섬웨어의 실행과정은 [그림 7]과 같다. 멀버타이징 또는 취약한 웹 브라우저를 통해 실행되는 Loader.dll은 Payload.dll을 언패킹하고, 감염 대상 PC의 프로세스에 DLL(Dynamic Link Library) 인젝션을 수행한다. 인젝션된 Payload.dll은 PC의 시스템 언어가 홍콩, 마카오, 중국, 싱가포르, 대만, 한국, 브루나이, 말레이시아 환경이 아닌 경우 PC를 감염시키지 않고 프로세스를 종료한다.

감염 대상 PC이면 파일의 중복 암호화를 방지하기 위해 뮤텍스를 생성한다. 분석에 사용한 샘플의 경우 'dyaaghemy' 문자열을 뮤텍스명으로 사용한다. 공격자가 만든 pseudo random 함수를 사용하여 0~9 또는 a~z의 문자를 무작위로 추출하여 총 19자리로 구성된 감염자의 고유 식별자(Victim ID)를 생성한다. 파일 암호화

〔표 7〕 constant 배열 값

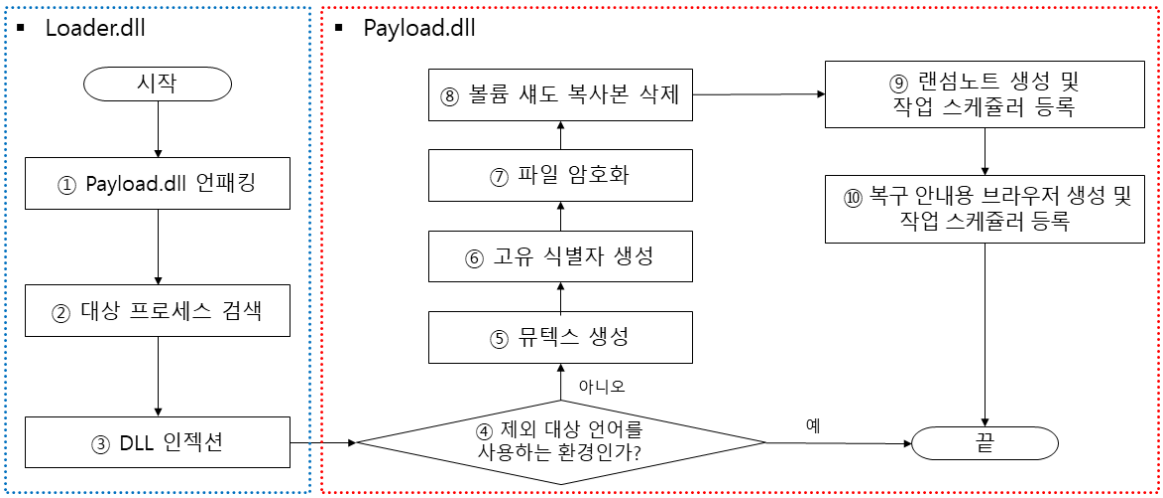
배열	값
constant[5]	0xb54ba0
constant[4]	0x6f910
constant[3]	0x44a8
constant[2]	0x2a4
constant[1]	0x1a
constant[0]	0x1

속도를 증가시키기 위해 감염 PC의 논리 드라이브 개수만큼 쓰레드를 생성하여 병렬적으로 파일 암호화를 수행한다.

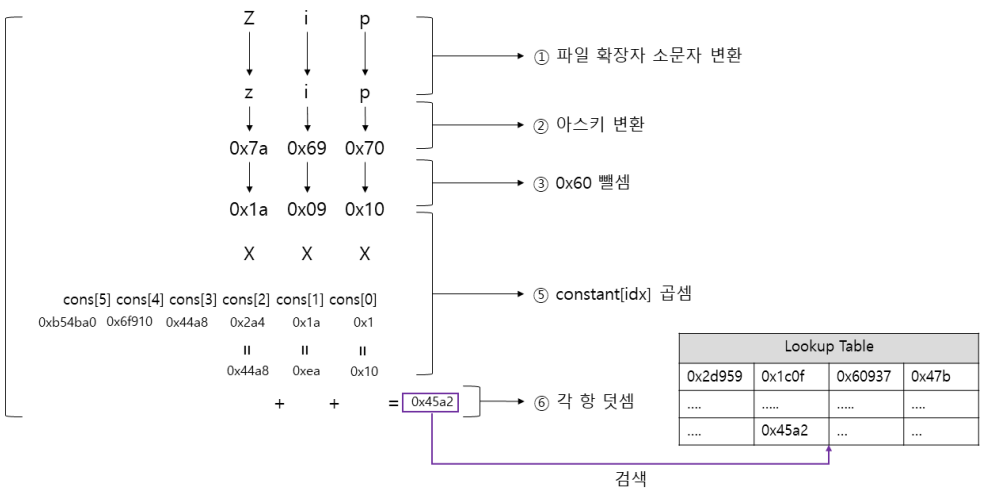
Magniber2 랜섬웨어는 암호화 대상 폴더 내 파일의 경우, 파일 확장자를 확인하여 암호화 수행 여부를 결정한다. 단순 문자열 비교가 아닌 룩업 테이블(Lookup Table)을 사용하여 확인한다.

먼저 파일 확장자의 문자열 길이가 7보다 크면 해당 파일을 암호화하지 않는다. 반대의 경우, 파일 확장자의 문자열을 차례대로 하나씩 선택하여 소문자로 변환한다. 변환된 문자열을 아스키코드 값으로 변경하고, 변경된 값에서 0x60 값을 뺀다. 그리고 그 값을 [표 7]의 메모리에 적재된 constant 배열의 각 문자 자릿수에 대응하는 요소와 곱한다.

위의 과정을 파일 확장자의 모든 문자열에 대해 수행한 후 획득한 모든 값의 합을 계산한다. 총합과 룩업 테이블을 비교하여 일치하는 경우에는 파일 암호화를 수행하고, 일치하지 않으면 파일 암호화를 수행하지 않는다.



〔그림 7〕 Magniber2 랜섬웨어 실행과정



〔그림 8〕 암호화 대상 파일 확장자 비교 과정

예를 들어, 파일 확장자가 Zip이면 모든 문자열을 소문자로 변환한다. 그리고 z, i, p 각각에 해당하는 아스키코드 값(0x7a, 0x69, 0x70)에 0x60 값을 뺀 값(0x1a, 0x09, 0x10)을 constant 배열의 요소들과 각각 곱하고 그 값을 더한다(constant[2] × 0x1a + constant[1] × 0x09 + constant[0] × 0x10 = 0x2a4 × 0x1a + 0x1a × 0x09 + 0x1 × 0x10 = 0x45a2). 위의 과정을 수행하여 얻은 결과 값을 [그림 8]과 같이 메모리에 존재하는 룩업 테이블과 비교한다. 파일 암호화가 완료되면, 암호화된 파일의 원본 확장자 뒤에 'dyaaghemy' 문자열이 추가된다.

파일 암호화가 완료된 후, 윈도우 백업 기능을 사용하지 못하도록 레지스트리 경로와 값을 생성한다. 그리고 이벤트 뷰어를 실행하여 생성된 레지스트리 값이 동작하도록 한다. 생성된 레지스트리 값은 볼륨 섀도 복사본을 삭제하는 명령어이다. 레지스트리 경로는 HKCU\Software\Classes\mscfile\shell\open\command이고 명령어는 cmd.exe /c "%SystemRoot%\system32\wbem\wmic shadow copy delete"이다.

그리고 %Public% 경로에 랜섬노트를 생성하고, 특정 시간마다 자동으로 실행되도록 작업 스케줄러에 등록한다. 고유 식별자, 확장자명 등의 문자열을 이용하여 복구 안내 페이지 URL을 생성하고, 특정 시간마다 복구 안내 페이지가 자동으로 나타나게 하도록 작업 스케줄러에 등록한다[28].

2.6.2. 암호화 과정

Magniber2 랜섬웨어는 파일 암호화 시 사용할 AES-128-CBC 암호 알고리즘의 암호 키와 IV를 생성한다. 공격자가 만든 pseudo random 함수를 사용하여 한 개의 문자(0~9 또는 a~z)를 만드는 과정을 16회 반복하여 16 bytes 크기의 암호 키 및 IV를 생성한다. 암호 키와 IV는 모든 파일마다 새로 생성한다.

pseudo random 함수는 난수 생성기로 생성한 랜덤한 값인 global_feedback 값과 현재의 tick count 값을 연산한 값으로 global_feedback 값을 갱신한다. tick count 값은 getTickCount 함수를 사용하여 생성하는데, 이 함수는 시스템이 부팅된 순간부터 현재까지 흐른 시간을 1/1000초 단위로 DWORD(32비트)를 반환하는 함수이다. 마지막으로 global_feedback 값을 최솟값과 최댓값 사이의 값으로 보정하고 반환한다. 최솟값과 최댓값은 pseudo random 함수의 입력 값으로 받은 변수이다.

Magniber2 랜섬웨어는 파일 암호 키와 IV를 연접한 값을 공격자의 공개 키를 이용하여 RSA-2048 알고리즘으로 암호화한다. 공격자의 공개 키는 CryptAcquire Context 함수를 사용하여 CSP(Cryptographic Service Provider)를 생성하고, 공격자의 RSA 공개 키 BLOB을 import 하여 가져온다. 해당 랜섬웨어는 파일 암호화 시 생성한 16 bytes 크기의 파일 암호 키를 Key BLOB 구조로 구성한다. 먼저, CryptImportKey 함수를 사용하여

변환된 Key BLOB을 CSP에 import 한다. 그리고 암호 알고리즘 사용에 필요한 부가정보를 설정하는 CryptSetKeyParam 함수를 사용하여 IV를 구성한다. Key BLOB을 구성한 후, 파일 암호화를 수행한다. 한 번에 0x100000bytes 크기만큼 데이터를 읽어와 AES-128-CBC로 암호화하고, 이 과정을 반복하여 파일의 모든 데이터를 암호화한다. 파일 암호 키와 IV는 파일마다 새로 생성되며, 공격자의 공개 키로 RSA-2048 암호화되어 EOF(End of File)에 저장된다.

Magniber2 랜섬웨어는 Windows의 CryptoAPI를 사용하여 파일과 파일 암호 키를 암호화한다. 암호키 사용이 완료되면 생성된 키 핸들을 삭제하는 기능의 CryptDestroyKey 함수를 사용하여 키 핸들을 메모리에서 삭제한다. 그리고 CSP의 핸들을 삭제하는 기능의 Crypt ReleaseContext 함수를 사용하여 메모리에서 CSP 정보를 삭제한다. 또한, 프로세스의 Heap 핸들을 가져오는 기능을 가진 GetProcessHeap 함수와 동적으로 할당된 메모리를 해제하는 기능을 가진 HeapFree 함수를 사용하여 현재 실행 중인 Magniber2 랜섬웨어의 프로세스 Heap 영역을 찾아 사용된 암호 키와 IV를 삭제한다[29].

2.6.3. 복구 가능성

Magniber2 랜섬웨어는 파일 암호화 시 AES-128-CBC 암호 알고리즘을 사용하고, 파일 암호 키 및 IV 암호화 시 RSA-2048을 사용한다. 파일 암호화 시 사용하는 암호 키와 IV는 pseudo random 함수를 사용하여 파일마다 생성된다. pseudo random 함수는 시스템 부팅 이후부터 현재까지의 tick count를 seed로 사용한다. 그리고 16 bytes 크기의 암호 키를 생성하기 위해 getTickCount 함수를 16번 실행한다.

결과적으로, 파일 암호 키를 재현하기 위해서는 모든 파일에 대해 호출된 tick count 값을 알아야 하며, 모든 tick count 값을 추측하는 것은 어렵다. 그리고 CryptDestroyKey 함수와 HeapFree 함수를 사용하여 메모리 내에 남아있는 파일 암호 키와 IV를 삭제하기 때문에, 메모리 분석을 통해 암호 키를 복구할 수 없다. 따라서 파일 복구를 하기 위해서는 공격자의 RSA 개인키가 필요하므로 복호화는 불가능하다[30].

Ⅲ. 결 론

본 논문에서는 CLOP,LooCipher 등 총 6종 랜섬웨어의 암호기능 및 복구 가능성을 분석하였다. 파일 암호화 시 AES와 같은 대칭 암호 키를 주로 사용하며, RC4, ChaCha20과 같은 스트림 암호를 사용하기도 한다. 그리고 사용한 암호 키는 공격자의 RSA 공개 키로 암호화하여 감염자가 이에 대응하는 개인 키를 알 수 없는 경우 복구가 불가능하도록 하였다. 향후 암호화 과정이 복잡하고 다양한 암호 알고리즘을 결합하여 사용하는 랜섬웨어들이 등장할 것으로 보인다. 따라서 랜섬웨어의 실행과정, 암호기능을 분석하여 복구 가능성을 판단할 필요가 있으며, 랜섬웨어가 사용한 암호 알고리즘 동향을 파악할 필요가 있다.

참 고 문 헌

[1] “KISA”, 2020년 7대 사이버 공격 전망, 2019

[2] “ITWORLD”, http://www.itworld.co.kr/news/143876

[3] “Ahnlab”, https://asec.ahnlab.com/1236

[4] “estsecurity”, https://www.estsecurity.com/securityCenter/malwareReport/view/1598

[5] 박해룡, 김기문, 김대운, 이영주, “CLOP 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-5, 2019

[6] 박해룡, 김기문, 김대운, 이영주, “CLOP 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 6-9, 2019

[7] 박해룡, 김기문, 김대운, 이영주, “CLOP 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 10, 2019

[8] “Bleeping Computer” https://www.bleepingcomputer.com/news/security/new-loocipher-ransomware-spreads-its-evil-through-spam/

[9] 박해룡, 김기문, 김대운, 이영주, “LooCipher 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-4, 2019

[10] 박해룡, 김기문, 김대운, 이영주, “LooCipher 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 5-6, 2019

[11] 박해룡, 김기문, 김대운, 이영주, “LooCipher 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 7-8, 2019
[12] “Checkmal blog”, http://blog.checkmal.com/221264669194
[13] “ZDNet”, https://www.zdnet.com/article/new-phobos-ransomware-exploits-weak-security-to-hit-targets-around-the-world/
[14] 박해룡, 김기문, 김대운, 이영주, “Phobos 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 1-9, 2019
[15] 박해룡, 김기문, 김대운, 이영주, “Phobos 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 10-12, 2019
[16] 박해룡, 김기문, 김대운, 이영주, “Phobos 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 13, 2019
[17] “Bleeping Computer”, https://www.bleepingcomputer.com/news/security/new-jcry-ransomware-spreads-its-evil-through-spam/
[18] 박해룡, 최은영, 김기문, 김대운, 이영주, “JCry 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-5, 2019
[19] 박해룡, 최은영, 김기문, 김대운, 이영주, “JCry 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 6-8, 2019
[20] 박해룡, 최은영, 김기문, 김대운, 이영주, “JCry 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 9, 2019
[21] “Checkmal Blog”, http://blog.checkmal.com/221546878712
[22] “Ahnlab Blog”, https://asec.ahnlab.com/1233
[23] 박해룡, 최은영, 김기문, 김대운, 이영주, “Maze 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-3, 2019
[24] 박해룡, 최은영, 김기문, 김대운, 이영주, “Maze 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 4-7, 2019
[25] 박해룡, 최은영, 김기문, 김대운, 이영주, “Maze 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 8, 2019
[26] “Malwarebytes Labs”, https://blog.malwarebytes.com/threat-analysis/2017/10/magniber-ransomware-exclusively-for-south-koreans/
[27] “Ahnlab Blog”, https://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?seq=27312
[28] 박해룡, 최은영, 김기문, 김대운, 이영주, “Magniber2 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-12, 2019
[29] 박해룡, 최은영, 김기문, 김대운, 이영주, “Magniber2 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 13-18, 2019
[30] 박해룡, 최은영, 김기문, 김대운, 이영주, “Magniber2 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 19, 2019

〈저자소개〉

이 영 주 (Yeong Ju Lee)
정회원
2017년 2월 : 전남대학교 지리학과 졸업
2018년 6월~현재 : 한국인터넷진흥원 차세대암호인증팀 주임연구원
<관심분야> 암호 구현, 랜섬웨어